establishment of a successful connection (i.e., communication link) between the two. Applications that can successfully authenticate against a server are said to be "trusted" applications with that server. Typically, an application downloaded or retrieved from a server is entrusted by that server.

[**0082**] For example, a client application may successfully authenticate with a server that contains resources such as e-mail services, printing resources, and other shared networked peripherals. However, the application's access rights may be limited (e.g., by the network administrator, or the application's programmer) to using the e-mail services only. Thus, while a client application may be able to successfully authenticate against a server, it may not have the permission to use all the resources available on that server.

[**0083**] The opposite can be also true. While an application may have the access rights to use a resource on a server, it may be unable to successfully authenticate against that server. For example, an application retrieved from a main network server may have access rights to all resources on that network, including resources available on computers other than the server. But due to network security schemes (i.e., firewalls) and safety measures embedded in the browser architecture, the application may not be able to authenticate against a computer where a desired resource resides. Thus while the aforesaid safeguards are helpful to protect a network's resources from unauthorized access, they are also limiting and undesirable where they prohibit an application from accessing resources that are otherwise available to it.

[**0084**] Embodiments of the invention provide an environment in which an application can indirectly access services or resources available on servers that it cannot directly authenticate against. **FIG. 4** is a block diagram illustrating the various components of one or more embodiments of the invention. Accordingly, the invention comprises application **410**, client **430**, communication links **440** and **445**, network **450**, server **460**, resource **470**, and web server **480**.

[**0085**] Client **430** includes a browser within which application **410** is executed. Application **410** can be retrieved from a server, such as web server **480**, when the browser parses a document written in HTML or other languages (e.g., VRML, XML, SGML, etc.) identified on web server **480**, for example. Application **410** is a trusted application to web server **480**, such that it can successfully establish communication link **445** with web server **480** and can access its resources.

[**0086**] In addition to resources available on web server **480**, application **410** may also need to access resources available on other servers on network **450**, such as server **460**. To access those resources (i.e., resource **470**) application **410** needs to submit a request to the server that is linked to those resources. In embodiments of the invention, requests submitted by application **410** are processed and sent through client **430**. Client **430** acts as the execution vehicle for application **410** and may contain virtual machine **435**. Virtual machine **435** is able to provide a level of abstraction and an independent execution environment for application **410** so that it can run on any platform such as UNIX, Windows, or other operating systems.

[**0087**] **FIG. 5** is a flow diagram illustrating a method by which application **410** submits a request to resource server **460**, according to one or more embodiments of the invention. At step **510**, application **410**'s request is submitted to resource server **460**, after being converted to the proper format by client **430**. At step **520**, it is determined whether application **410** is a trusted application to server **460**. If so, application **410** establishes communication link **440** with server **460**, and at step **530** application **410**'s request for access to resource **470** is submitted to server **460**, using either the UDP or the TCP protocol. Once application **410**'s request is processed by server **460**, at step **535**, client application **410** receives a response from server **460**, via communication link **440**.

[**0088**] If at step **520** application **410** fails to successfully authenticate against resource server **460** (i.e., because it was not retrieved from that server), then it cannot directly submit the request to server **460**. Hence, an alternate route is needed so that application **410** can indirectly access resource **470**. In one or more embodiments of the invention, at step **540**, client **430** identifies web server **480**, the server against which application **410** can successfully authenticate (i.e., the server from which application **410** was retrieved), and submits the request to that server, via communication link **445**, using the HTTP/HTTPS protocol.

[**0089**] Web server **480** is a gateway on network **450** that can indirectly route application **410**'s requests to server **460**. Since application **410** is a trusted application, it can successfully submit its requests to web server **480**. Additionally, since web server **480** and resource server **460** are both members of network **450**, the two servers can communicate free from any limitations. These communication limitations can be, generally, imposed by network **450**'s firewall security measures or the browser's access constraints, developed to prohibit unauthorized access by external entities to network **450**.

[**0090**] Requests submitted by application **410** to web server **480** are processed by servlet **490**. Servlet **490** is a program code that can be written in the Java programming language, or other programming languages, and can access a resource server that can satisfy application **410**'s request. Servlet **490** can be invoked by application **410** via a name or a URL, for example. In one or more embodiments of the invention, based on the type and the nature of requests submitted by application **410**, at step **550**, a search is performed to locate servlet **490**.

[**0091**] At step **560**, it is determined whether servlet **490** is found on web server **480**. If servlet **490** is not found then an error occurs. This error is processed at step E (i.e., the user or the application is notified that the request cannot be processed, because access to the resource has been denied). Alternatively, if servlet **490** is found, then at step **580** the request submitted by application **410** is directed to servlet **490**. Servlet **490** acts as a proxy by routing requests and responses between application **410** and server **460**.

[**0092**] For example, one of servlet **490**'s function is to act as a conduit (or a "tunnel") between client application **410** and server **460**. **FIG. 6** is a flow diagram illustrating the manner in which servlet **490** operates, according to one or more embodiments of the invention. At step **610**, the request submitted by application **410** is processed by servlet **490** to determine whether application **410** is authorized to access resource **470**, as requested. This authorization is typically based on application **410**'s access rights, as decided by the programmer of the application, for example.